Flow to Interface the Camera Sensor

1) Interface the camera sensor correctly while the board is off
2) Turn on the board
3) Go to Vivado and flash with this bitstream : icarus v2.2.bit
4) Go to /vega-tools/utils/eth-transfer and do ./send.sh bbl.bin riscv.dtb
5) Open fresh terminal and do minicom trisul32
6) Bootloader… do root root and continue
7) Set up eth0 and enp1s0 everytime :) steps given below

> On board :
>
> > ip link set eth0 up
> >
> > ip addr add 192.168.1.20/24 dev eth0
>
> On Laptop :
>
> > sudo ip addr flush dev enp1s0
> >
> > sudo ip addr add 192.168.1.10/24 dev enp1s0
> >
> > sudo ip link set enp1s0 up

8) Write the capture code in c, and to access the address, review the image in next page…. (since camera is using i2c + spi, the image only describes flow for giving address for spi)
9) Scp from laptop the compiled capture binary

```
/home/amrut/VEGA/trisul32_buildroot/buildroot/output/host/bin/riscv32-linux-gcc
/home/amrut/Downloads/<whatever>.c -o <whatever> -static

/home/amrut/VEGA/trisul32_buildroot/buildroot/output/host/bin/riscv32-linux-strip
chunked_capture #(for stripping GDB headers)

ssh-keygen -f '/home/amrut/.ssh/known_hosts' -R '192.168.1.20'

scp -O <compiled_binary> root@192.168.1.20:/root/
```

10) Run the binary (ensure you strip the debug GDB headers)
11) SCP the image.bin back
12) Run the python script as provided in /camera in git

```
# --- SPI 0 (IMU: Gyro/Accel) -> Mapped to Pmod JC ---
set_property -dict { PACKAGE_PIN U12 IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_0_ss_io[0] }]; # JC[1] (CS)
set_property -dict { PACKAGE_PIN V12 IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_0_io0_io }];   # JC[2] (MOSI)
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_0_io1_io }];   # JC[3] (MISO)
set_property -dict { PACKAGE_PIN V11 IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_0_sck_io }];   # JC[4] (SCK)

# --- SPI 1 (Camera/Aux) -> Mapped to Standard ChipKit SPI Header ---
set_property -dict { PACKAGE_PIN C1  IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_ss_io[0] }]; # CK_SS
set_property -dict { PACKAGE_PIN H1  IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_io0_io }];   # CK_MOSI
set_property -dict { PACKAGE_PIN G1  IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_io1_io }];   # CK_MISO
set_property -dict { PACKAGE_PIN F1  IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_sck_io }];   # CK_SCK

## ------------------------------------------------------------
## SD CARD (SPI MODE) -> Mapped to ChipKit Outer Digital Header (Pins 0-3)
```
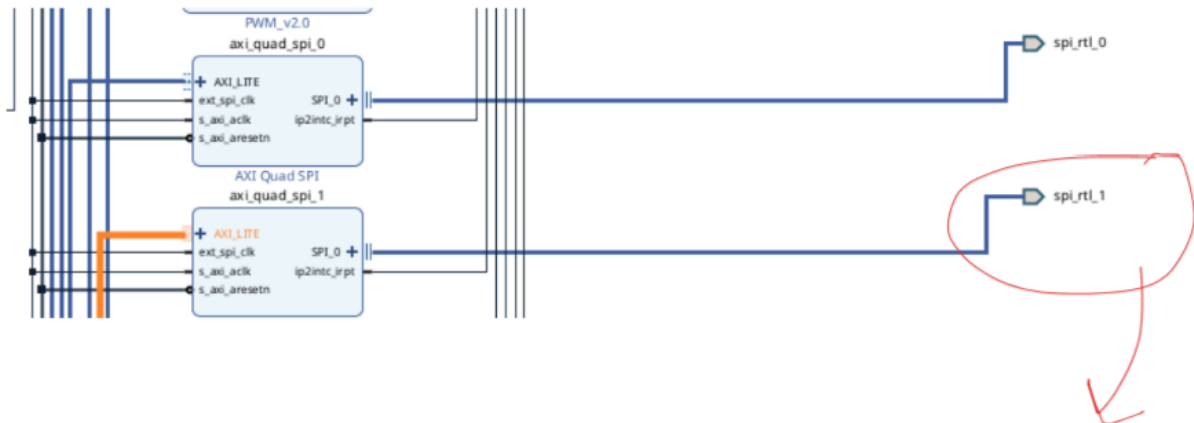

/axi_gpio_0/S_AXI	S_AXI	Reg	0x1008_0000		512K	▾	0x100F_FFFF
/axi_iic_0/S_AXI	S_AXI	Reg	0x1011_6000		4K	▾	0x1011_6FFF
/axi_quad_spi_0/AXI_LITE	AXI_LITE	Reg	0x1011_4000		4K	▾	0x1011_4FFF
/axi_quad_spi_1/AXI_LITE	AXI_LITE	Reg	0x1011_5000		4K	▾	0x1011_5FFF
/axi_quad_spi_2/AXI_LITE	AXI_LITE	Reg	0x1011_9000		4K	▾	0x1011_9FFF
/axi_uart16550_0/S_AXI	S_AXI	Reg	0x1000_1000		4K	▾	0x1000_1FFF
/axi_uartlite_0/S_AXI	S_AXI	Reg	0x1011_1000		4K	▾	0x1011_1FFF
/axi_uartlite_1/S_AXI	S_AXI	Reg	0x1011_2000		4K	▾	0x1011_2FFF
/axi_uartlite_2/S_AXI	S_AXI	Reg	0x1011_3000		4K	▾	0x1011_3FFF
/flight_math_unit_v1_0_0/s00_axi	s00_axi	reg0	0x1011_A000		4K	▾	0x1011_AFFF
/mig_7series_0/memmap	S_AXI	memaddr	0x8000_0000		256M	▾	0x8FFF_FFFF

To interface the camera sensor
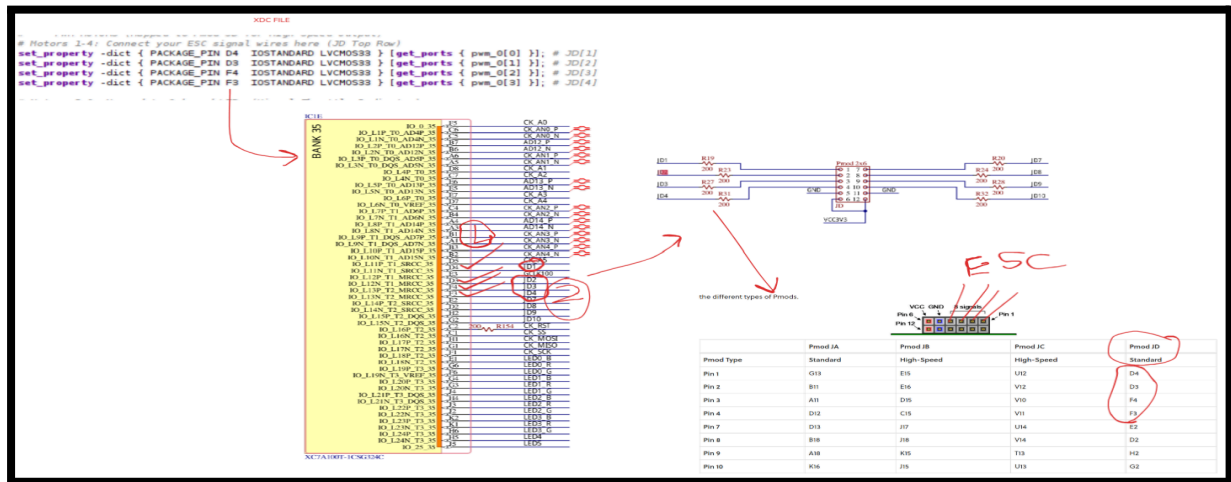
To interface ArduCam

| cs	c1	J6 pin 5
mosi	h1	J6 pin 4
miso	g1	J6 pin 1
sck	f1	J6 pin 3
gnd	Gnd	J7 gnd
vcc	3.3/5v	J7 vcc 3.3 -> 5v
sda	sda	sda
scl	scl	scl

--- SPI 1 (Camera/Aux) -> Mapped to Standard ChipKit SPI Header ---

set_property -dict { PACKAGE_PIN C1 IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_ss_io[0] }]; # CK_SS

set_property -dict { PACKAGE_PIN H1 IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_io0_io }]; # CK_MOSI

set_property -dict { PACKAGE_PIN G1 IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_io1_io }]; # CK_MISO

set_property -dict { PACKAGE_PIN F1 IOSTANDARD LVCMOS33 } [get_ports { spi_rtl_1_sck_io }]; # CK_SCK
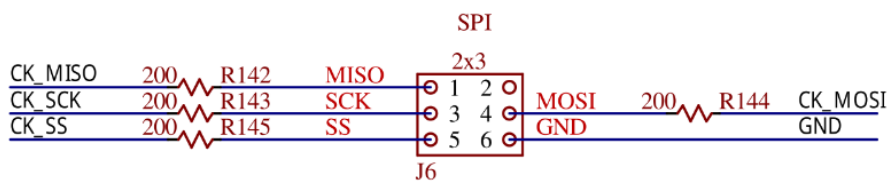
Why the board might not have internet………

Reasons:

1. FPGA Linux has **no Ethernet interface (`eth0`)**

2. No DHCP client

3. No default route

4. No DNS
5. Probably **no SSH server running**

Extra Information :) ---->

(In hardware and software, a magic number is a special, fixed value used for identification or control, often appearing as unexplained literals in code (e.g., 37px for positioning) or as file signatures (e.g., %PDF for PDFs) that uniquely identify file types or protocols, helping systems quickly recognize data formats. While useful for defining system parameters, they can make code harder to maintain, so they are often replaced by named constants for clarity, especially in hardware design and digital forensics

PNG File: Starts with 89 50 4E 47 (hex) which is ASCII for ".PNG".

JPEG File: Starts with FF D8 FF E0 (hex).

Why They Matter

Identification: Quickly identify file types or data formats without parsing the whole file.

Control: Dictate system behavior, like clock speeds in hardware designs.

Security: Vital in cybersecurity for malware detection and file validation.

Maintainability: Poorly documented magic numbers make code confusing; defining them as constants improves clarity)

Our dts (device tree) file

```
/dts-v1/;

/ {
  #address-cells = <2>;

  #size-cells = <2>;

  compatible = "ucbbar,spike-bare-dev";

  model = "ucbbar,spike-bare";

  chosen {

    stdout-path = &SERIAL0;

    bootargs = "console=hvc0 earlycon";

  };

  cpus {

    #address-cells = <1>;

    #size-cells = <0>;

     timebase-frequency = <10000>;

    CPU0: cpu@0 {

      device_type = "cpu";

      reg = <0>;

      status = "okay";

      compatible = "riscv";

      riscv,isa = "rv32ima";

      mmu-type = "riscv,sv32";
```

```dts
            clock-frequency = <40000000>;


            CPU0_intc: interrupt-controller {

                #interrupt-cells = <1>;

                interrupt-controller;

                compatible = "riscv,cpu-intc";

            };

        };

    };

    memory@80000000 {

        device_type = "memory";

        reg = <0x0 0x80000000 0x0 0x10000000>;


    };

    soc {

        #address-cells = <2>;

        #size-cells = <2>;

        compatible = "ucbbar,spike-bare-soc", "simple-bus";

        ranges;

        clint@20010000 {

            compatible = "riscv,clint0";

            interrupts-extended = <&CPU0_intc 3 &CPU0_intc 7>;
```

```dts
        reg = <0x0 0x20010000 0x0 0xc0000>;

};


ethernet@20030000 {

        device_type = "network";

        compatible = "cdac,cdac_mac";

        interrupt-parent = <&plic0>;

        interrupts = <9>;

        reg = <0x0 0x20030000 0x0 0x100>;

    };


plic0: interrupt-controller@20010000 {

 #interrupt-cells = <1>;

 compatible = "cdac,plic-1.0.0";

 reg = <0x0 0x20010000 0x0 0x4000000>;

 riscv,ndev = <32>;

 interrupt-controller;

 interrupts-extended = <&CPU0_intc 9>;

 };


 SERIAL0: ns16550@10001000{
```

```
        compatible = "ns16550a";

    clock-frequency = <40000000>;

    reg = <0x0 0x10001000 0x0 0x100>;

    reg-io-width = <4>;

    reg-shift = <2>;

    current-speed = <115200>;


};
spi_clock:spi_clock{

        #clock-cells = <0>;

        compatible = "fixed-clock";

        clock-frequency = <40000000>;

    };
    spi0: spi@10000600 {

        compatible = "cdac,spi";

        clock-names = "spi_clock";

        clocks = <&spi_clock>;

        interrupt-parent = <&plic0>;

        interrupts = <6>;

        reg = <0x0 0x10000600 0x0 0x100 >;

        #address-cells = <1>;

        #size-cells = <0>;

        spidev@0 {
```

```
            compatible = "cdac,spidev";

            reg = <0>;

            spi-max-frequency = <25000000>;

        };

    };

    gpio0: gpio-controller@10080000 {

        compatible = "cdac,cdac-gpio";

        reg = <0x0 0x10080000 0x0 0x50000>;

        gpio-base = <0>;

        gpio-controller;

        #gpio-cells = <2>;

    };

    gpio1: gpio-controller@10180000 {

        compatible = "cdac,cdac-gpio";

        reg = <0x0 0x10180000 0x0 0x50000>;

        gpio-base = <16>;

        gpio-controller;

        #gpio-cells = <2>;

    };

    i2c0: i2c@10000800 {

            compatible = "cdac,mdp-i2c";

        clock-frequency = <100000>;

        system-frequency = <40000000>;
```

```dts
        #address-cells = <1>;

        #size-cells = <0>;

        reg = <0x0 0x10000800 0x0 0x100>;

        interrupt-parent = <&plic0>;

        interrupts = <8>;

        status = "okay";

};
    i2c1: i2c@10000900 {

        compatible = "cdac,mdp-i2c";

        clock-frequency = <100000>;

        system-frequency = <40000000>;

        #address-cells = <1>;

        #size-cells = <0>;

        reg = <0x0 0x10000900 0x0 0x100>;

        interrupt-parent = <&plic0>;

            interrupts = <4>;

        status = "okay";


};
pwm_clock:pwm_clock{

    #clock-cells = <0>;

    compatible = "fixed-clock";

    clock-frequency = <40000000>;
```

```
        };
        pwm0: pwm@10400000 {

            compatible = "cdac,cdac-pwm";

            clock-names = "pwm_clock";

                clocks = <&pwm_clock>;

            reg = <0x0 0x10400000 0x0 0x100>;

            #pwm-cells = <3>;


        };
    };
};
```